New MAA XML Schema

An overview

JANUARY 2019

Contents

Contents

Current MAA schema problems and solutions adopted

What we saw in the DES XML specification	What we have done
The DES XML node order does not reflect the order of fields in the form. **Example(s):** The node *maa:scientific-advice* (Section 3) appears in the document before the node *maa:declaration* (First section in the PDF)	Reorganized the order of the nodes in the XML document so that it matches the order visually perceived in the PDF document
Some nodes in XML DES have names which corresponds to the UI control that allows its edition and does not reflect their business meaning. **Example(s):** Nodes *maa:yes*, *rdm:selected* or *maa:Manu-device-checkbox*	Changed the name of these nodes, so that it relates to the business meaning of the data it contains.
There are cases of high-level nodes that have the most part of its children nodes empty and cannot be filled from the PDF, resulting in an unnecessary overhead in size and complexity of the XML document. **Example(s):** The node *maa:contact-pharmaco-vigilance* has its children nodes *rdm:admin-office* and *rdm:manu-facility* with all their descendants empty with no possibility of being filled via the PDF form.	Removed unnecessary nodes (i.e.: all the nodes that cannot be filled using the PDF form)

Current MAA schema problems and solutions adopted (Cont.)

What we saw in the DES XML specification	What we did
Attributes of some XML nodes are equal to “” and cannot be modified from the PDF. **Example(s):** Attribute *is_significant-benefit* of node *maa:orphan-designation*	Removed all attributes in the XML. Plain (scalar) data are just contained as node values (i.e.: “<node>data</node>”)
There are some attributes, not related to the business but with the adobe technology , defined in the XFA schema and with namespace *http://www.xfa.org/schema/xfa-data/1.0/*. **XFA** is a proprietary family of XML specifications (see Wikipedia) used also by Adobe. **Example(s):** Attribute *xfa:APIVersion*	Removed all attributes in the XML, hence also proprietary ones
Node naming conventions are not uniform. Sibling nodes are not named following the same rules and notation. **Example(s):** • Node *maa:centralised-procedure* corresponds to section 1.1.1 of the PDF • Node *maa:section1-5-1* (sibling of the above) corresponds to section 1.5.1	Named all nodes consistently. Children of *form* node have names that reflect corresponding section number in the PDF document. All nodes of lower level have been given names that reflect their business meaning: • *“section-1”* for Section 1 node • *“centralisedProcedure”*, for Section 1.1 node

Current MAA schema problems and solutions adopted (Cont.)

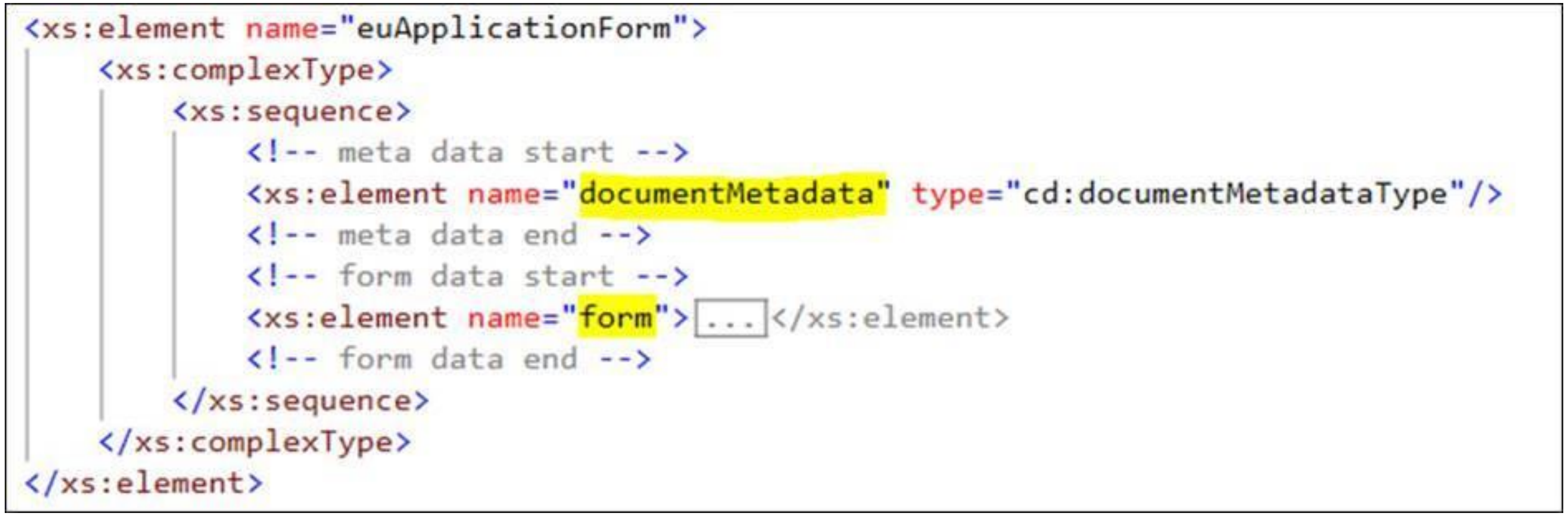
What we saw in the DES XML specification	What we did
Nodes aren’t named consistently. Different use of lower/uppercase characters, uderscores (“_”) and dashes (“-“) in node names. **Example(s):** Nodes *maa:subject_to_prescription*, *maa:Device-identification*, *rdm:loc-modifiedDate*, *maa:not-subject-medical-prescription*	Adopted Camel case naming convention (Wikipedia) uniformely, while preserving DES names, where possible. Dash (“-“) character is used only in section names and annex- names, in order to separate contained numbers. **Examples:** *subjectToPrescription, deviceIdentificarion, notSubjectMedicalPrescription, annex-2*
Repeatable nodes of different type share the same parent. **Example(s):** Nodes of type *maa:Manufacture-contact-details* (repeatable – Section 2.2.4.2) are siblings of nodes *maa:Device-identification* (repeatable too – Section 2.2.4.1)	Each collection of repeatable nodes is included in a node having same name, but “pluralized” and with “Collection” suffix **Example:** *manufacturerContactDetailsCollection* node contains a collection of *manufacturerContactDetail* nodes

Current MAA schema problems and solutions adopted (Cont.)

What we saw in the DES XML specification	What we did
Some nodes are used improperly. **Example(s):** If *“Vaccine antigen master file”* option is selected in the PDF, a *rdm:vamf* node is generated with all its subnodes empty except one (*rdm:is-vamf-issued* with a value of “1”). Data of all VAMF entries are located in following *rdm:vamf* (first entry in second node, second entry in third and so on), so that reading of first node has to be skipped in order to retrieve data.	Created a node called *isVamfIssued* wich indicates whether the option is selected or not. The node has a sibling called *vamfs* that contains the collection of *vamf* nodes, one corresponding to each real VAMF entry.

Contents

New schema structure overview

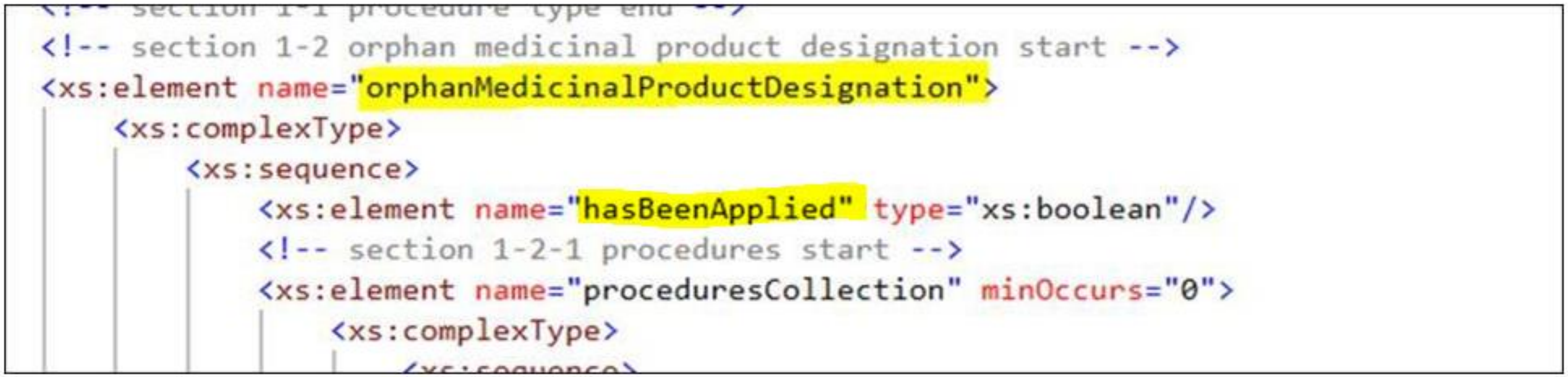


```
<xs:element name="euApplicationForm">
    <xs:complexType>
        <xs:sequence>
            <!-- meta data start -->
            <xs:element name="documentMetadata" type="cd:documentMetadataType"/>
            <!-- meta data end -->
            <!-- form data start -->
            <xs:element name="form">...</xs:element>
            <!-- form data end -->
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Remarks:

- Document has root node *“euApplicationForm”* with two children:
  - *documentMetadata:* domain, version, creation date
  - *form:* the eAF data
- Only two namespaces:
  - maa:http://www.eaf.com/maa/ (instances data namespace)
  - cd:http://www.eaf.com/dictionary/ (dictionary namespace)
- No external namespaces (~~xfa=http://www.xfa.org/schema/xfa-data/1.0/~~)

New schema structure overview (Cont.)

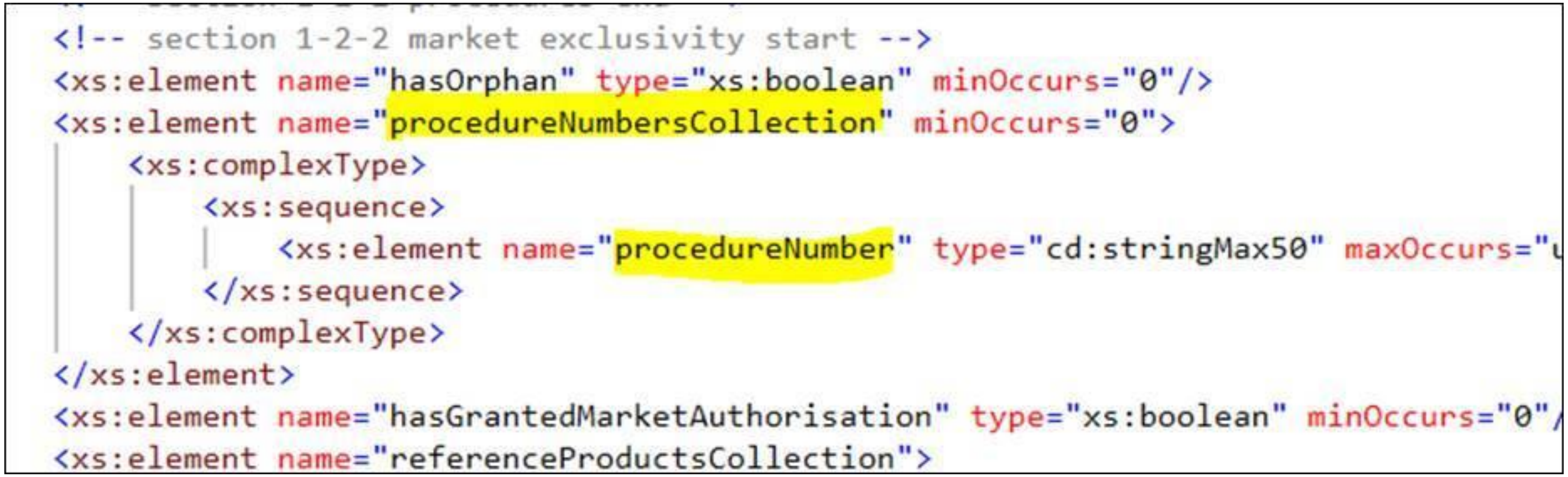


```
<!-- section 1-1 procedure type end -->
<!-- section 1-2 orphan medicinal product designation start -->
<xs:element name="orphanMedicinalProductDesignation">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="hasBeenApplied" type="xs:boolean"/>
            <!-- section 1-2-1 procedures start -->
            <xs:element name="proceduresCollection" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
```

Remarks:

- Enforced consistent naming conventions (use of *Camel Case* notation)
- Boolean nodes (yes/no) have names that suggest their nature (begin with “has” “is” “was”,…)
- Used XSD schema to define/enforce:
    - Data types
    - Be mandatory/optional
    - Cardinality

New schema structure overview (Cont.)

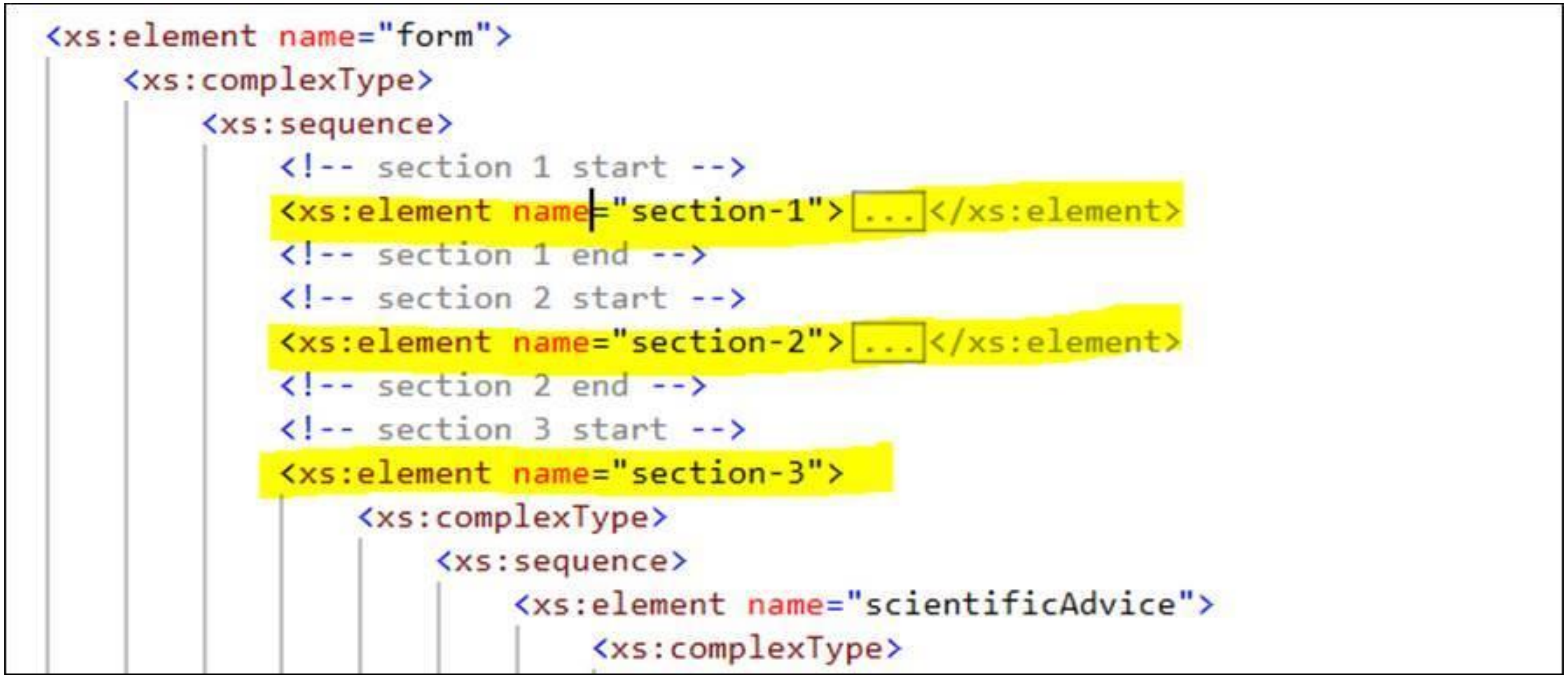


```
<!-- section 1-2-2 market exclusivity start -->
<xs:element name="hasOrphan" type="xs:boolean" minOccurs="0"/>
<xs:element name="procedureNumbersCollection" minOccurs="0">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="procedureNumber" type="cd:stringMax50" maxOccurs="u
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="hasGrantedMarketAuthorisation" type="xs:boolean" minOccurs="0"/
<xs:element name="referenceProductsCollection">
```

Remarks:

- Collections of repeteable nodes always contained in an **exclusive** parent
- Parent name of nodes *xxx* is named *xxxsCollection* (plural form + “Collection” suffix)
- **exclusive** = no other nodes contained in it

New schema structure overview (Cont.)

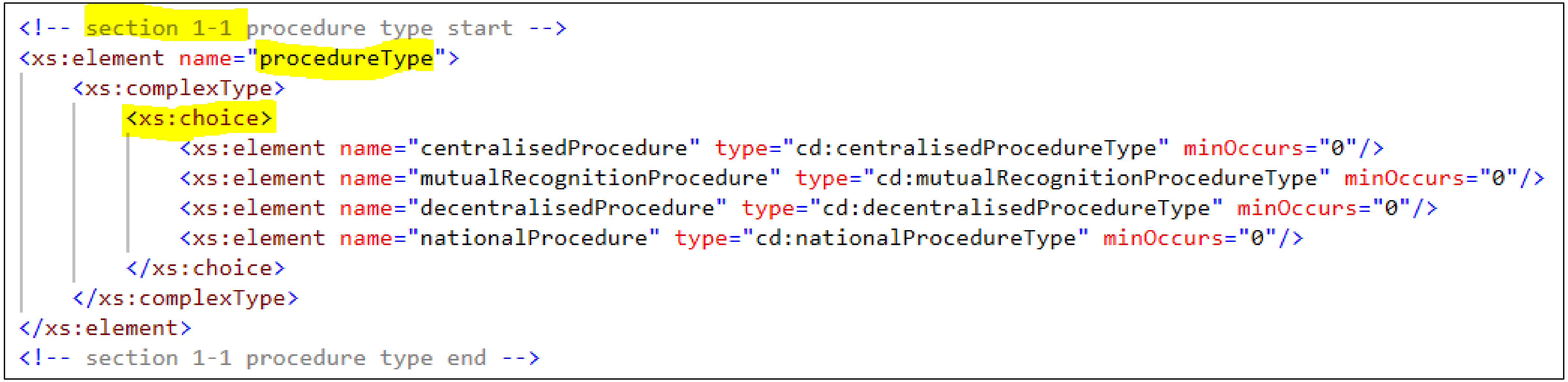


```
<xs:element name="form">
    <xs:complexType>
        <xs:sequence>
            <!-- section 1 start -->
            <xs:element name="section-1">...</xs:element>
            <!-- section 1 end -->
            <!-- section 2 start -->
            <xs:element name="section-2">...</xs:element>
            <!-- section 2 end -->
            <!-- section 3 start -->
            <xs:element name="section-3">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="scientificAdvice">
                            <xs:complexType>
```

Remarks:

- First level children of *"form"* have names that reflect their position in current PDF structure

New schema structure overview (Cont.)

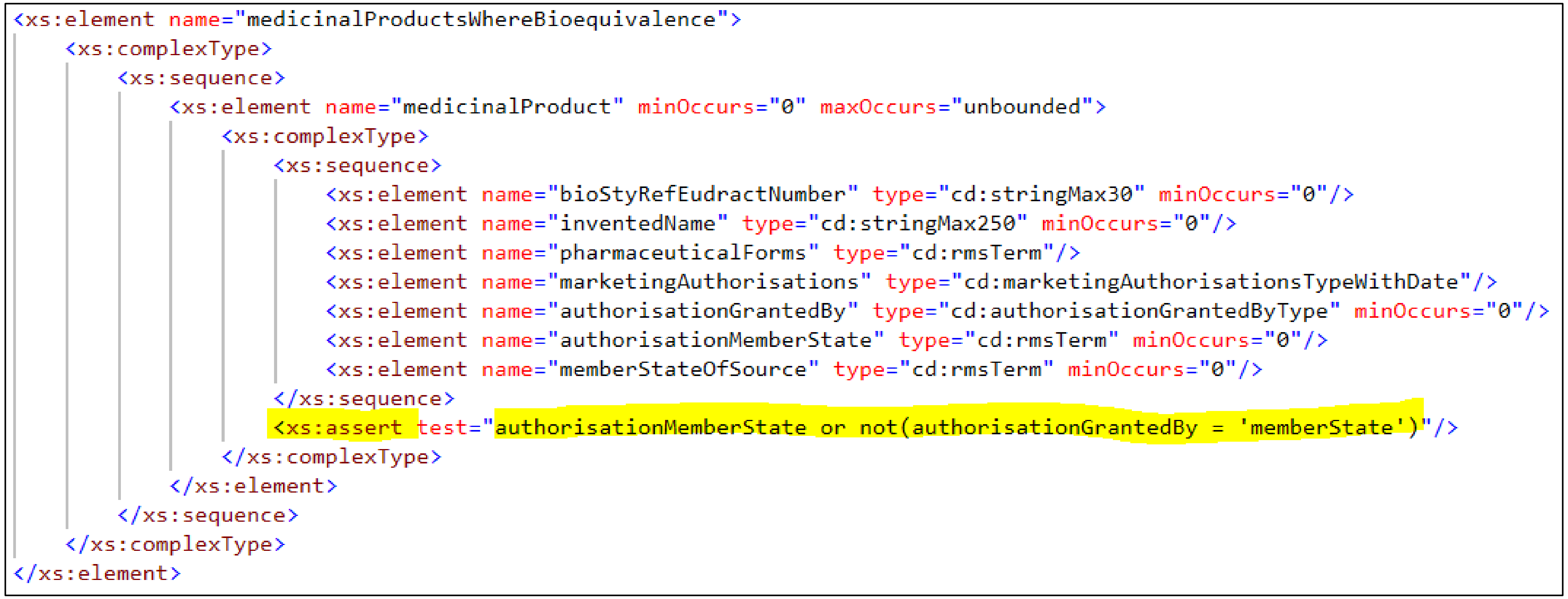


```
<!-- section 1-1 procedure type start -->
<xs:element name="procedureType">
    <xs:complexType>
        <xs:choice>
            <xs:element name="centralisedProcedure" type="cd:centralisedProcedureType" minOccurs="0"/>
            <xs:element name="mutualRecognitionProcedure" type="cd:mutualRecognitionProcedureType" minOccurs="0"/>
            <xs:element name="decentralisedProcedure" type="cd:decentralisedProcedureType" minOccurs="0"/>
            <xs:element name="nationalProcedure" type="cd:nationalProcedureType" minOccurs="0"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
<!-- section 1-1 procedure type end -->
```

Remarks:

- Children of 2nd level (or higher) have names that reflect their business meaning (e.g.: *centralizedProcedure*)
- Use of *xs:choice* element to limit document size and avoid inconsistencies (if it is a *centralizedProcedure*, no *nationalProcedure* contents exist)
- Use of types and deferred node definitions (in the dictionary)

New schema structure overview (Cont.)

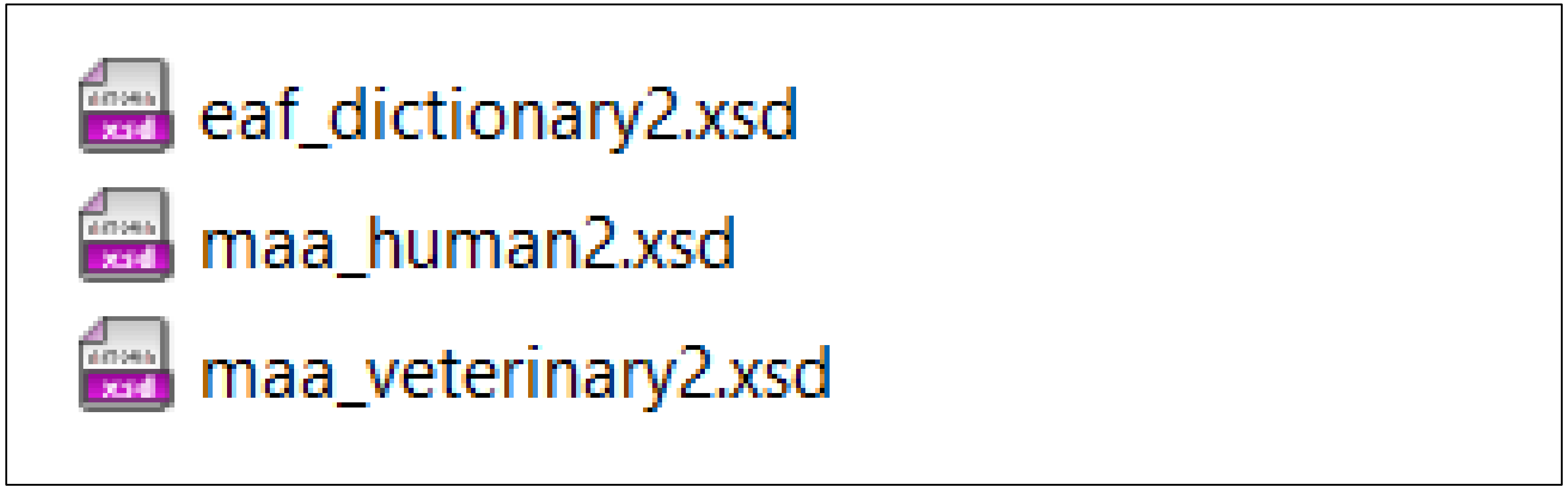


```
<xs:element name="medicinalProductsWhereBioequivalence">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="medicinalProduct" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="bioStyRefEudractNumber" type="cd:stringMax30" minOccurs="0"/>
                        <xs:element name="inventedName" type="cd:stringMax250" minOccurs="0"/>
                        <xs:element name="pharmaceuticalForms" type="cd:rmsTerm"/>
                        <xs:element name="marketingAuthorisations" type="cd:marketingAuthorisationsTypeWithDate"/>
                        <xs:element name="authorisationGrantedBy" type="cd:authorisationGrantedByType" minOccurs="0"/>
                        <xs:element name="authorisationMemberState" type="cd:rmsTerm" minOccurs="0"/>
                        <xs:element name="memberStateOfSource" type="cd:rmsTerm" minOccurs="0"/>
                    </xs:sequence>
                    <xs:assert test="authorisationMemberState or not(authorisationGrantedBy = 'memberState')"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Remarks:

- Use of *xs:assert* elements to enforce complex business rules compliance (**Note:** XSD schema **version 1.1** used)
- Assertions defined after element fields enumeration, so the element definition is always self-contained

New schema structure overview (Cont.)



eaf_dictionary2.xsd

maa_human2.xsd

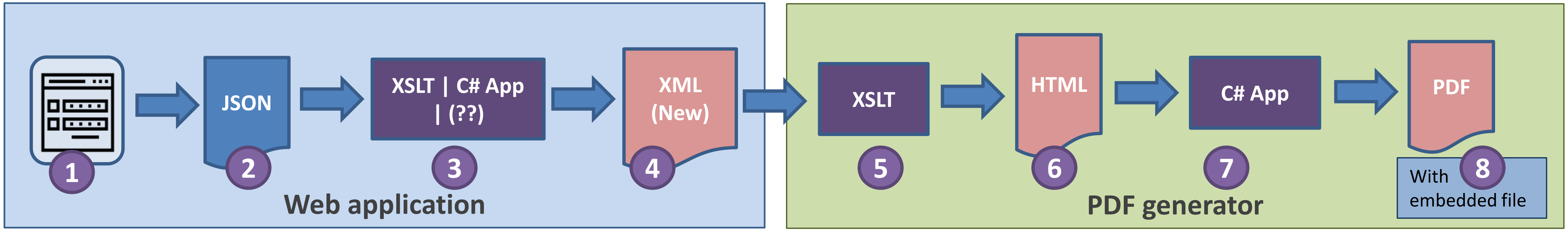
maa_veterinary2.xsd

Remarks:

- Three XSD schemas:
  - **maa_human2.xsd** (eAF of **human** domain)
  - **maa_veterinary2.xsd** (eAF of **veterinary** domain)
  - **eaf_dictionary.xsd** (**common** dictionary with type definitions for **both domains**)
- Version should be **2.0** (no backward compatibility with 1.2x.* versions)

Contents

PDF generation in CESP project



Data flow:

1. Data is inputted by users in a web form
2. Internally, a JSON representation of the data is created
3. Using a transformation mechanism (XSLT or coded-solution) a XML document is generated
4. This XML conforms to new XSD specification (thus, XSD schemas can be used to perform input validation)
5. An XSLT transformation is used for generating an HTML representation of the eAF
6. This XML has a visual appearance similar to current PDF Form (in its **1.23.1.1** version)
7. A coded solution will implement the transformation to PDF of the HTML document. Third-party libraries will be used (IronPDF + iTextSharp)
8. The generated PDF will have the eAF XML document attached (natively embedded). Additionally, the JSON document can be embedded as well